Subject: Re: Generating errors
Posted by Braedley on Thu, 16 Nov 2006 17:47:41 GMT
View Forum Message <> Reply to Message

Sorry for the confusion. The program will make changes to parts of it's state structure. In doing so, the original event sent by the user can no longer be performed until the user triggers the event again, after the problem has been fixed, so the error handler returns out of the event handler and waits for another event. If the user does totally **** things up, there is another option for him, but he still does have to reload all the data he was working on. And I was already using the same type of error handler that you suggested.

David Fanning wrote:
> Braedley writes:
>
>> The error would be in the state structure, and would be due to it not
>> being updated properly after some other error (that I still have to
>> devise a runtime fix for). The thing is that both n and (*state).m
>> both become useless for performing the action that the user wanted to
>> perform after the problem was fixed. In all likelihood, the original
>> value of (*state).m is different from what it should have been, and it
>> then becomes impossible to retrieve the value that it should have been
>> with 100% accuracy before the fix is implemented. Since I don't want
>> to perform an action on (*state).m (that is not easy to undo) without
>> being sure that it's the action that the user wants, I'm forced to
>> return after the problem is fixed for future use.
>
> Huh? Not sure I'm following everything here... Too many
> fingers pointing in too many directions, I guess.
>
> In most of the widget programs I've ever written, the
> user can get himself out of it, if the widget program
> just keeps running. That is to say, there is *usually*
> some way the user can fix the problem for themselves.
> So, I usually just tell the user what the problem was,
> and keep the program alive for them to figure out the
> solution. An error handler like this usually works
> well for me:
>
>  Catch, theError
>  IF theError NE 0 THEN BEGIN
>  Catch, /Cancel
>  void = Error_Message()
>  ; Whatever clean-up is needed in THIS module. For
>  ; example, put the info structure back in TLB, etc.
>  RETURN
>  ENDIF

> 
> If you have errors that you *can't* recover from (this is
> what your explanation seems to imply to me), then I think
> you are basically hosed. Destroy your TLB and start over. :-)
> 
> Cheers,
> 
> David
> 
> 
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

---